Примена техника машинског учења у статичкој верификацији софтвера

Лазар Ранковић 1099/2016
Немања Мићовић 1085/2016
Урош Стегић 447/2016

Математички факултет

12. мај 2017.

Садржај

Мотивација

- Важност процеса верификације софтвера

Увод

- Верификација?
- Основни приступи
  - Динамичка верификација
  - Статичка верификација
- Халтинг проблем *(енгл. halting problem)*

Методи статичке верификације

- Апстрактна интерпретација
- Проверавање модела
- Симболичко израчунавање

Машинско учење

Увод

- Грађење и употреба алгоритама који генерализују
- Писање програма који се прилагођавају
- Статистичко учење из података

Машинско учење

Мотивација

- Алгоритамски ~~нерешиви~~ тешко решиви проблеми
- Задаци које човек лако решава
- Предиктивна анализа

Машинско учење

Класификација

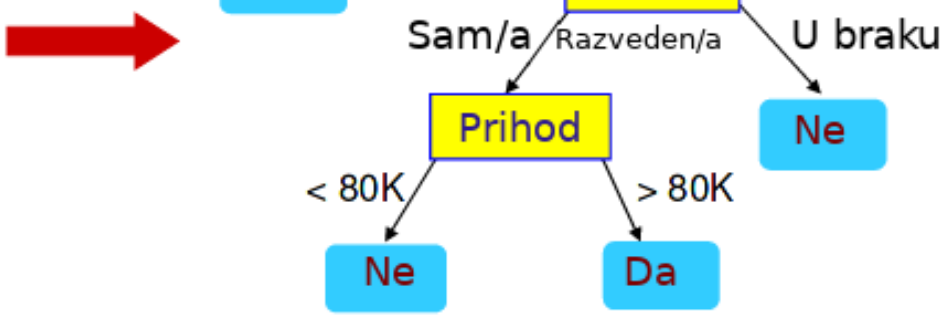
- Препознавање објеката на фотографији
- Разврставање непожељне поште
- Класификација стања програма

Машинско учење

Стабло одлучивања

Tid	Povraćaj Novca	Bračni status	Prihod	Prevara
1	Da	Sam/a	125K	Ne
2	Ne	U braku	100K	Ne
3	Ne	Sam/a	70K	Ne
4	Da	U braku	120K	Ne
5	Ne	Razveden/a	95K	Da
6	Ne	U braku	60K	Ne
7	Da	Razveden/a	220K	Ne
8	Ne	Sam/a	85K	Da
9	Ne	U braku	75K	Ne
10	Ne	Sam/a	90K	Da

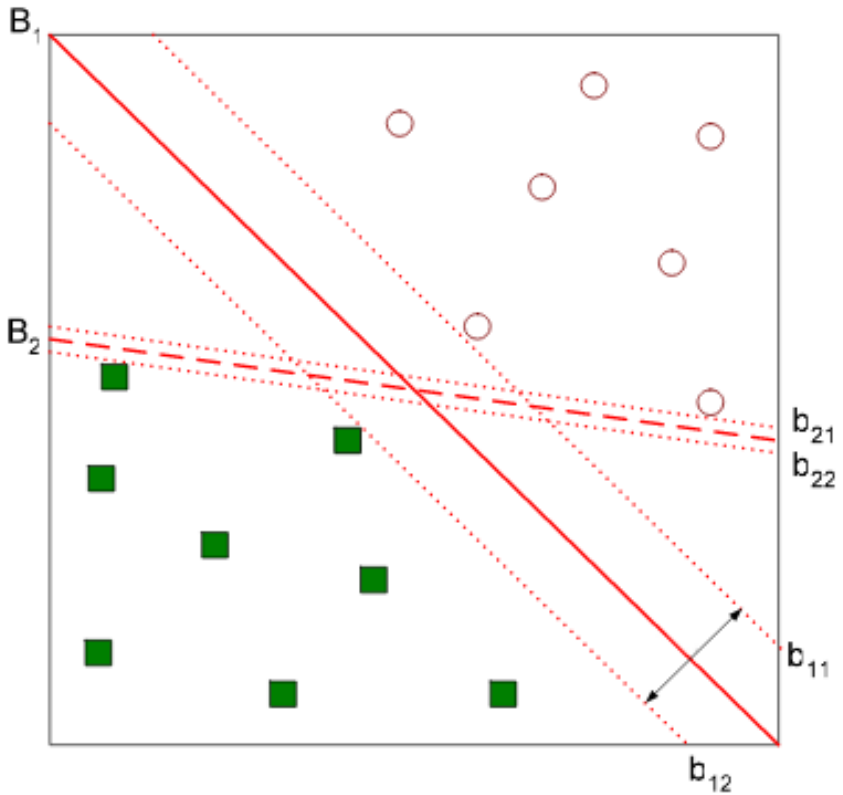
Trening podaci



Model: Stablo odlučivanja

Машинско учење

Метода потпорних вектора



Машинско учење

Још мало мотивације

- Магија у НП проблемима
- Машинско учење на белом коњу

Проналажење интерполанти - потпорни вектори

- Делимо скуп стања програма на скупове A и B
- A садржи вредности x и y након линија 1, 2, 3
- B садржи вредности x и y након линија 4, 5, 6 и 7

```
funkcija primer()
{
1:     x = y = 0;
2:     while (e)
3:     { x++; y++; }
4:     while (x != 0)
5:     { x--; y--; }
6:     if (y != 0)
7:     greska();
}
```

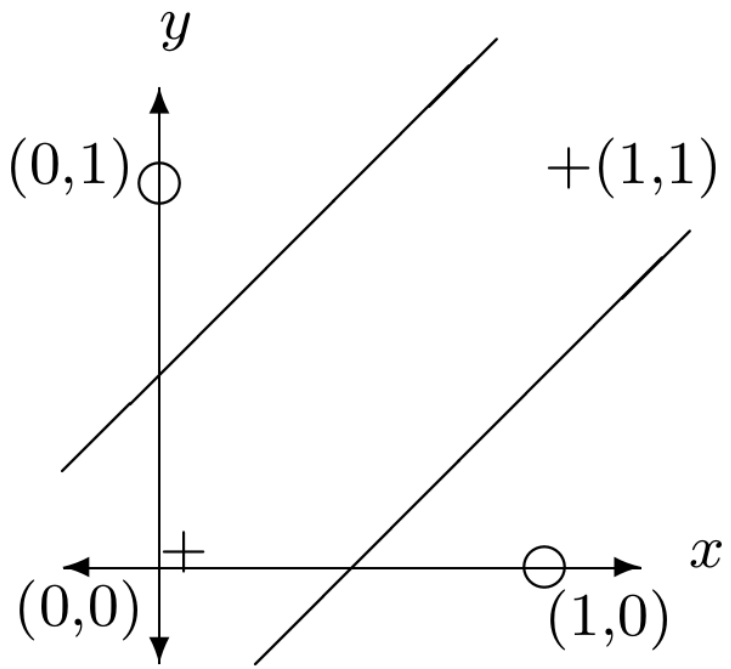
- Интерполанта је доказ да су скупови A и B дисјунктни
- Доказивач теорема рачуна вредности за променљиве x и y
- Добија се скуп инстанци над којима се може тренирати модел машинског учења

$$A \equiv x_1 = 0 \wedge y_1 = 0 \wedge ite(e,\ x = x_1 + 1 \wedge y = y_1 + 1,\ x = x_1 \wedge y = y_1)$$

$$B \equiv ite(x = 0,\ x_2 = x \wedge y_2 = y,\ x_2 = x - 1 \wedge y_2 = y - 1) \wedge x_2 = 0 \wedge \neg(y_2 = 0)$$

Проналажење интерполанти - потпорни вектори

- Модел добијен применом метода потпорних вектора
- Приказане праве одговарају једначинама:
  $p1 : 2y = 2x + 1$
  $p2 : 2y = 2x - 1$
- Можемо извести интерполанту
  $2y \leq 2x + 1 \wedge 2y \geq 2x - 1$
- Инваријанта $x = y$ се може добити транслирањем правих што ближе позитивним инстанцама



Грађење класификатора нетачне инваријанте

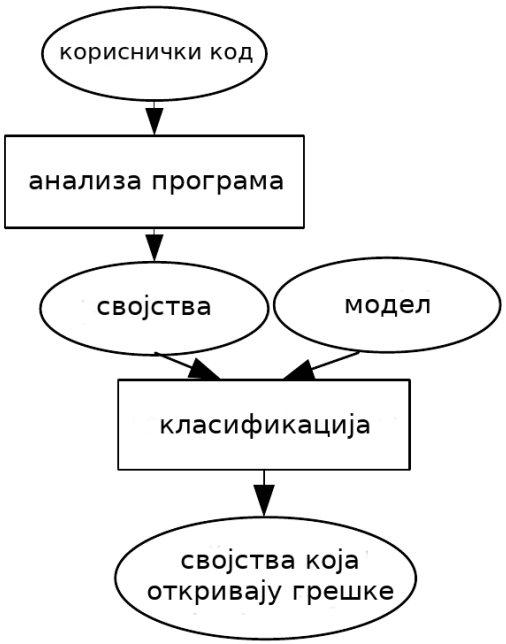
- Рангирају се својста програма по вероватноћи за стварање грешке
- Циљ је програмеру понудити листу својстава које потенцијално треба проверити

```
// Vraca sortiranu kopiju argumenta
double[] bubble_sort(double[] in) {
    double[] out = kopiraj_niz(in);
    for (int x = out.duzina - 1; x >= 1; x--)
    // donja granica treba da bude 0, ne 1
    for (int y = 1; y < x; y++)
        if (out[y] > out[y+1])
            razmeni(out[y], out[y+1])
    return out;
}
```

Својства	Открива грешку?
$\texttt{out}[1] \leq \texttt{in}[1]$	Да
$\forall\, \texttt{i}\colon \texttt{in}[\texttt{i}] \leq 100$	Не
$\texttt{in}[0] = \texttt{out}[0]$	Да
$\text{size}(\texttt{out}) = \text{size}(\texttt{in})$	Не
$\texttt{in} \subseteq \texttt{out}$	Не
$\texttt{out} \subseteq \texttt{in}$	Не
$\texttt{in} \neq \text{null}$	Не
$\texttt{out} \neq \text{null}$	Не

Грађење класификатора нетачне инваријанте

- Користи се DAIKON динамички детектор инваријанти који може да проверава:
  - уређење ($x \leq y$)
  - опсег ($a \leq x \leq b$)
  - линеарне везе ($z = ax + by + c$)
  - и друге
- Добијена својства се кодирају у векторе и примењује се метода машинског учења



Својства	Једначина				Тип променљиве			# v	Резултат
	$\leq$	$=$	$\neq$	$\subseteq$	int	double	array		
out[1] $\leq$ in[1]	1	0	0	0	0	1	0	2	19
$\forall$ i: in[i] $\leq$ 100	1	0	0	0	0	1	0	1	16
in[0] = out[0]	0	1	0	0	0	1	0	2	15
size(out) = size(in)	0	1	0	0	1	0	0	2	13
in $\subseteq$ out	0	0	0	1	0	0	1	2	12
out $\subseteq$ in	0	0	0	1	0	0	1	2	12
in $\neq$ null	0	0	1	0	0	0	1	1	10
out $\neq$ null	0	0	1	0	0	0	1	1	10
Тежине модела	7	3	2	1	4	6	5	3	

Закључак

- Машинско учење може имати добру примену у верификацији софтвера
- Примена машинског учења у верификацији софтвера се активно развија и истражује
- Добијени резултати су барем упоредиви са традиционалним методама верификације софтвера

Хвала на пажњи